Stop-Word Removal Algorithm for Arabic Language

Dr. Riyad Al-Shalabi  
Chairman of CS Dept.  
Yarmouk University/Irbid-Jordan  
shalabi@yu.edu.jo

Dr. Ghassan Kanaan  
Chairman of CIS Dept.  
Yarmouk University/Irbid-Jordan  
ghassank@yu.edu.jo

Pr. Jihad M. Jaam  
Computer Science Department  
Qatar University, P.O. Box 2713  
Doha – Qatar  
jaam@qu.edu.qa

Dr. Ahmad Hasnah  
Computer Science Department  
Qatar University, P.O. Box 2713  
Doha – Qatar  
hasnah@qu.edu.qa

Eyard Hilat  
Graduate Student  
Yarmouk University/Irbid-Jordan

Abstract
In this paper, we have designed and implemented an efficient stop-word removal algorithm for Arabic language based on a Finite State Machine (FSM). An efficient stop-word removal technique is needed in many natural language processing applications such as: spelling normalization, stemming and stem weighting, Question Answering systems and in Information Retrieval systems (IR). Most of the existing stop-word removal techniques base on a dictionary that contains a list of stop-word, it is very expensive, it takes too much time for searching process and required too much space to store these stop-words.

The new Arabic removal stop-word technique has been tested using a set of 242 Arabic abstracts chosen from the proceedings of the Saudi Arabian National Computer conferences, and another set of data chosen from the holy Q’uran, and it gives impressive results that reached approximately to 98%.

Introduction
Stop-words has been recognized since the earliest days of information retrieval [16]. Stop-words are words that are evenly distributed in a document corpus and are among the most frequent words in a language. Some of the words contained in a document are considered to be worthless as index terms. Searching an index for one of these terms might include all records in the database regardless of their relevancy. These words make up a large portion of the text of most documents. Removal of such words from the index saves space and does not damage retrieval effectiveness [18].

Stop-words belong to several word groups such as conjunctions, propositions, adverbs etc, and often referred to as function words. Most often used stop-word lists for English are rather small, consisting of 200-400 words, due to morphological richness of the language, the list contains all possible morphological variants of each stop-word.

In written text, some words are very common and have no additional meaning to the actual content of the text, and have little or nothing to say about the text itself. Prepositions, conjunctions, nouns and articles are examples of such words. A lot of processing time and working memory can be saved if the words that do not contribute to the actual content of the corpus are removed. The percentage that the corpus is being decreased is often about 70 – 75 % compared with before the reduction was done. This is done by filtering
the term list with so called "stop-lists" also called
syncretomorphic or non-context bearing words [15].

When dealing with Arabic text (especially in its
electronic form), one has to be aware of the common
mistakes and miss-spelling that occur in the text. For
example, although the word "ل" starts with one form
of alif (alif with hamza above "ل"), it is common to find it
spelled as "ـ". Other common mistakes are using haa
("أ") instead of taa marbutah("إ") at the end of the word
and vice-versa. Also for most of the Arabic stop-words
there is a combination of prefixes may appear. Not all
combinations of the prefixes are allowed. For example,
for the word "لم" the prefixes that allowed are ("ع", "م", and "ف"). Only "ف" and "ل" are valid others are not (e.g., "ل").
Another conjunctive prefix "ع" is a possible prefix for most Arabic texts.

Algorithm description

Most systems that removes stop-words use a
file or a dictionary that contains a list of stop-words, to
check a word if it is a stop-word or not, several
techniques are used, one is to use a traditional way
to search the dictionary, compare each word until the stop-
word found or reaches the end of the file, and this will
take time O(n) in sequential search and O(log n) using
binary search. Second, would use hashing with different
versions, and this will need more space to be efficient.
Third is the dynamic stop-word removing, that is to
consider the words that has a high frequency as a
stop-words and append them to the stop-list.

In this paper we proposed a new technique to
speed up the process of removing of stop-word in the
Arabic text using FSM. We break down the stop-list
and represent each word in the FSM, then implement
the FSM in main memory as a state table that contains
293 rows (states) and 36 columns represents the following
Arabic letters: أ, إ, ب, تقث, ج, د, ف, ض, ط, ص, ي. The
algorithm of stop-word removal is as follows:
Step 0: input: Arabic text;
output: list of stop-words in the given Arabic
text;
get the first word from the text;
Step 1: remove any non-Arabic letter from the word,
and call it clean_word;
Step 2: Let w_length is the length of clean_word
If w_length < 2 then
clean word is a stop-word, move to
the next word, and goto step 1.
Let CntState = Counter = 1;
While Counter < w_length and
CntState > 0 do
CntState = the value of
intersection in the state table of the
CntState row value
with the letter column in the
position Counter in the
clean_word.
Counter = Counter + 1;
End while
If CntState = 0 then goto step 3
If Counter > w_length and if CntState value is
a final state then

Counter = Counter + 1;
End while
If CntState = 0 then goto step 3
If Counter > w_length and if CntState value is
a final state then

clean_word is a stop-word, move to the
next word in the text, goto
Step 3: This step is required because of the nature of
the


Algorithm description

Most systems that removes stop-words use a
file or a dictionary that contains a list of stop-words, to
check a word if it is a stop-word or not, several
techniques are used, one is to use a traditional way
to search the dictionary, compare each word until the stop-
word found or reaches the end of the file, and this will
take time O(n) in sequential search and O(log n) using
binary search. Second, would use hashing with different
versions, and this will need more space to be efficient.
Third is the dynamic stop-word removing, that is to
consider the words that has a high frequency as a
stop-words and append them to the stop-list.

In this paper we proposed of new technique to
speed up the process of removing of stop-word in the
Arabic text using FSM. We break down the stop-list
and represent each word in the FSM, then implement
the FSM in main memory as a state table that contains
293 rows (states) and 36 columns represents the following
Arabic letters: أ, إ, ب, تقث, ج, د, ف, ض, ط, ص, ي. The
algorithm of stop-word removal is as follows:
Step 0: input: Arabic text;
output: list of stop-words in the given Arabic
text;
get the first word from the text;
Step 1: remove any non-Arabic letter from the word,
and call it clean_word;
Step 2: Let w_length is the length of clean_word
If w_length < 2 then
clean word is a stop-word, move to
the next word, and goto step 1.
Let CntState = Counter = 1;
While Counter < w_length and
CntState > 0 do
CntState = the value of
intersection in the state table of the
CntState row value
with the letter column in the
position Counter in the
clean_word;

Implementation

In our technique, we used a
deterministic Finite Automata (DFA) to
to check whether the word is a stop-word or
not, sample of a DFA that accepts the
words "لا", "كل", "كلما", "كلما", "كلما", and
"كلما" as a stop-word is shown in figure 1.
We translate the DFA into a state table data
structure, rows in the table represent the name of the states, where the underline
label means that this state is a final state,
and the columns represent the different
letters in the Arabic language, the values in
the table represents the next transition
states in the DFA for a given word, sample
of state table is shown in figure 2.
Figure 1. Sample of a DFA

The system was implemented using Visual Basic, sample of the output of the system is shown in figure 3.

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>9</td>
<td>11</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>8</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>14</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2. Sample of a state table

Conclusions

In this paper we designed and implemented an algorithm for removing the Arabic stop-words based on a deterministic finite machine, we created the stop-list of more than 1,000 words using several sources, one is the list compiled by Bonnie Glover Stalls, and Yaser Al-Onaizan, and another from translating English stop-lists to Arabic. We tested our system using 242 Arabic abstracts from the Proceedings of the Saudi Arabian National Computer conferences with a total number of 47897 words, we ran our system on a 550MHz PC, the system tooks 26 second and resulted a 12891 stop-words. We also ran the system on another set of data taken from the Holy Quraan of a total of 7050 Arabic words, and the system resulted 3235 stop-words. The accuracy of the system is reached to 98%.

References

[1] James Allan, NLP for IR, Natural Language Processing for Information Retrieval, Center for Intelligent Information Retrieval Department of Computer Science University of Massachusetts, Amherst.


[9] Dictionary-based Thai CLIR :Experimental Survey of Thai CLIR, Jaruskulchai Chuleerat, Department of Computer Science, Faculty of Science, Kasetsart University.


162, April 1963. References and Citations. [BibTeX entry]


Figure 3. sample output of the system